

10/528101  
PCT / IB 02 / 05 195

24. 12. 02

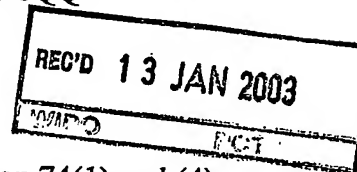


INVESTOR IN PEOPLE

The Patent Office  
Concept House  
Cardiff Road  
Newport  
South Wales  
NP10 8QQ

# **PRIORITY DOCUMENT**

SUBMITTED OR TRANSMITTED IN  
COMPLIANCE WITH RULE 17.1(a) OR (b)



I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.



*Andrew Gersey*

Signed

Dated 29 November 2002

The  
Patent  
Office

19SEP02 E749294-2 D00068

PO 0221674.5

Cardiff Road  
Newport  
Gwent NP9 1RH



0221674-5

# Request for grant of a patent

(See the notes on the back of this form. You can also get an explanatory leaflet from the Patent Office to help you fill in this form)

1. Your reference	800282/PRS/JN		
2. Patent application number (The Patent Office will fill in this part)	0221674.5		
3. Full name, address and postcode of the or of each applicant (underline all surnames)	NOKIA CORPORATION KEILALAHDENTIE 4 02150 ESPOO FINLAND		
Patents ADP number (if you know it)	7652217003		
If the applicant is a corporate body, give the country/state of its incorporation	FINLAND		
4. Title of the invention	LINKED AUTHENTICATION PROTOCOLS		
5. Name of your agent (if you have one)	PAGE WHITE & FARRER		
"Address for service" in the United Kingdom to which all correspondence should be sent (including the postcode)	54 DOUGHTY STREET LONDON WC1N 2LS UNITED KINGDOM		
Patents ADP number (if you know it)	1255003		
6. If you are declaring priority from one or more earlier patent applications, give the country and the date of filing of the or of each of these earlier applications and (if you know it) the or each application number	Country (if you know it)	Priority application number (if you know it)	Date of filing (day / month / year)
7. If this application is divided or otherwise derived from an earlier UK application, give the number and the filing date of the earlier application	Number of earlier application		Date of filing (day / month / year)
8. Is a statement of inventorship and of right to grant of a patent required in support of this request? (Answer 'Yes' if: a) any applicant named in part 3 is not an inventor, or b) there is an inventor who is not named as an applicant, or c) any named applicant is a corporate body	YES		

Enter the number of sheets for any of the following items you are filing with this form.  
Do not count copies of the same document

Continuation sheets of this form

Description	25
Claim(s)	6
Abstract	1
Drawing(s)	7

27

10. If you are also filing any of the following, state how many against each item.

Priority documents

Translations of priority documents

Statement of inventorship and right to grant of a patent (Patents Form 7/77)

Request for preliminary examination and search (Patents Form 9/77)

Request for substantive examination (Patents Form 10/77)

Any other documents  
(please specify)

11. I/We request the grant of a patent on the basis of this application.

PJ

Signature  
PAGE WHITE & FARRER

Date

18 September 2002

12. Name and daytime telephone number of person to contact in the United Kingdom

Philip R Slingsby 020 7831 7929

**Warning**

After an application for a patent has been filed, the Comptroller of the Patent Office will consider whether publication or communication of the invention should be prohibited or restricted under Section 22 of the Patents Act 1977. You will be informed if it is necessary to prohibit or restrict your invention in this way. Furthermore, if you live in the United Kingdom, Section 23 of the Patents Act 1977 stops you from applying for a patent abroad without first getting written permission from the Patent Office unless an application has been filed at least 6 weeks beforehand in the United Kingdom for a patent for the same invention and either no direction prohibiting publication or communication has been given, or any such direction has been revoked.

**Notes**

- If you need help to fill in this form or you have any questions, please contact the Patent Office on 0645 500505.
- Write your answers in capital letters using black ink or you may type them.
- If there is not enough space for all the relevant details on any part of this form, please continue on a separate sheet of paper and write "see continuation sheet" in the relevant part(s). Any continuation sheet should be attached to this form.
- If you have answered 'Yes' Patents Form 7/77 will need to be filed.
- Once you have filled in the form you must remember to sign and date it.
- For details of the fee and ways to pay please contact the Patent Office.

Patents Form 1/77

## LINKED AUTHENTICATION PROTOCOLS

This invention relates to authentication protocols.

Authentication protocols are used in communication networks to enable one entity in the network to satisfy another entity of its capacity to be provided with a service, for example the undertaking of a data transaction or the granting of access to a resource. For example, if a user's mobile telephone is to access an e-mail account on a remote server then the server may wish to authenticate the mobile phone so that it can be sure that it is making the contents of the account available to the correct entity. Similarly, the user of the mobile phone may want the phone to authenticate the server so that he can be sure that data that he provides to the server is not being intercepted by a third party who might be impersonating the server.

A number of authentication protocols have been developed.

The Extensible Authentication Protocol (EAP), described in [RFC2284], provides a standard mechanism for support of multiple authentication methods. Through the use of EAP, support for a number of authentication schemes may be added, including smart cards, Kerberos, Public Key, One Time Passwords, and others. Specific EAP/SIM and EAP/AKA methods have been defined for the purposes of the OWLAN system, which makes use of the GSM or UMTS authentication methods for the authentication of WLAN access and derivation of link keys for the protection of the WLAN link.

EAP is a general authentication protocol, designed to allow end-points to use multiple forms of authentication. EAP does not require the server (typically a PPP or IEEE 802 end-point) to authenticate the client itself, rather it allows the server to proxy authentication messages to a back-end authentication server, and inspect the packets to determine if the authentication was successful.

One of the goals of EAP is to enable the development of new authentication methods without requiring the deployment of new code on the local authentication server. As a result, the local server acts as a "passthrough", and need not understand specific EAP methods.

---

Since its deployment, a number of weaknesses in EAP have become apparent. These include lack of protection of the user identity or the EAP negotiation and no standardized mechanism for key exchange. For example, EAP/SIM and EAP/AKA solve these problems by making use of the specific features of SIM and USIM (AKA) authentication methods.

Recently IETF working groups have designed three new protocols to develop standard solutions. Two of them PEAP and TTLS, originating from the pppext working group, are intended to solve the same problem as EAP/SIM and EAP/AKA<sup>[1]</sup>, but independently of the specific EAP method. The idea in brief is to encapsulate the EAP protocol within TLS. The third protocol PIC is developed by the ipsra working group, and it develops a unilateral version of the ISAKMP authentication and key derivation protocol, within which the EAP protocol is encapsulated. All three protocols include a method of managing secret parameters for further use. In PEAP and TTLS keys are derived from the TLS master key, while in PIC the tunnel derived by simplified ISAKMP is used to transport security association from the local authentication server to the client.

By wrapping the EAP protocol within TLS, Protected EAP (PEAP) claims to provide user anonymity and built-in support for key exchange.

Figure 1 shows the relationship between the EAP peer (client), the network authentication server (NAS) and the backend authentication server. As described in the figure, the EAP conversation "passes through" the NAS on its way between the client and the backend authentication server. While the authentication conversation is between the EAP peer and the backend authentication server, the NAS and the

backend authentication server need to have established trust for the conversation to proceed.

In PEAP, the conversation between the EAP peer and the backend server is encrypted and integrity protected within a TLS channel, and mutual authentication is required between the EAP peer and the backend server.

As a result, the NAS does not have knowledge of the TLS master secret derived between the EAP Peer and the backend authentication server, and cannot decrypt the PEAP conversation. In order to providing keying material for link-layer ciphersuites however, the NAS does obtain the master session keys, which are derived from the TLS master secret via a one-way function.

Since EAP methods may not know the link layer ciphersuite that has been negotiated, it may not be possible for them to provide link layer ciphersuite-specific keys. In addition, attempting to provide such keys is undesirable, since it would require the EAP method to be revised each time a new link layer ciphersuite is developed. As a result, PEAP derives master session keys, which can subsequently be truncated for use with a particular link layer ciphersuite. PEAP does not discuss the format of the attributes used to communicate the master session keys from the backend authentication server to the NAS; examples of such attributes are provided in [RFC2548].

The operation of PEAP is as follows:

1. Establish TLS connection. The TLS record protocol provides a secure connection between the peer and the back-end authentication server
2. Authenticate TLS server. The TLS handshake protocol is used for server authentication
3. Authenticate user. The user of the peer authenticates by tunnelling another EAP mechanism (e.g. Generic Token Card) inside the EAP-TLS connection.

The back-end authentication server may have to contact another server to get the user authentication information validated.

---

4. Generate session keys. Using the TLS Pseudo-Random Function (PRF), the peer and the back-end server generate key material for use between NAS and peer.
5. (Transport session keys). The session key is transported from the server to the authenticator using e.g. Radius attributes and secure connection.

In a new internet draft, draft-buckley-pppext-eap-sim-gmm-00.txt, an application of PEAP to GSM authentication is presented. The architectural overview of this EAP method using SIM is depicted in Figure 2.

It should be noted that the network entity called "back-end server" in the description of PEAP, see Figure 1, has not at all the same role in the authentication protocol as the back-end server in PIC has. The PEAP back-end server is called the EAP Server in Figure 2 and its role is essentially the same as that of the TTLS AAA server of EAP-TTLS (figure 3) and the AS of PIC (figure 4).

The architectural view of EAP-TTLS is essentially the same as in PEAP, see Figure 3. EAP-TTLS claims to allow legacy password-based authentication protocols to be used against existing authentication databases, while protecting the security of these legacy protocols against eavesdropping, man-in-the-middle and other cryptographic attacks.

EAP-TTLS also allows the client and server to establish keying material for use in the data connection between the client and access point. The keying material is established implicitly between the client and server based on the TLS handshake.

When record layer security is instantiated at the end of a TLS handshake, a pseudo-random function (PRF) is used to expand the negotiated master secret, server

random value and client random value into a sequence of octets that is used as keying material for the record layer.

EAP-TTLS leverages this technique to create keying material for use in the data connection between client and access point. Exactly the same PRF is used to generate as much keying material as required, with the constant string set to "ttls keying material", as follows:

```
EAP-TTLS_keying_material = PRF(SecurityParameters.master_secret,  
                                "ttls keying material",  
                                SecurityParameters.client_random +  
                                SecurityParameters.server_random);
```

The master secret, client random and server random used to generate the data connection keying material must be those established during the TLS handshake. Both client and TTLS server generate this keying material, and they are guaranteed to be the same if the handshake succeeded. The TTLS server distributes this keying material to the access point via the AAA carrier protocol.

The PIC protocol is a method to bootstrap IPsec authentication via an "Authentication Server" (AS) and user authentication mechanisms (e.g., RADIUS). The client machine communicates with the AS using a key exchange protocol where only the server is authenticated, and the derived keys are used to protect the user authentication. Once the user is authenticated, the client machine obtains credentials from the AS that can be later used to authenticate the client in a standard IKE exchange with an IPsec-enabled security gateway. The later stage does not require user intervention.

The PIC protocol realizes this approach and secures it using simplified ISAKMP and IKE mechanisms. The protocol embeds EAP messages [RFC2284] in ISAKMP payloads to support multiple forms of user authentication. Once this user authentication succeeds, the client machine obtains from the AS credentials that can



later be used by the client to perform regular IKE authentication with an IPsec-enabled gateway. PIC defines several forms of credentials and can be extended to support others. Note that this document uses the term "credentials" for both digital certificates and shared secret keys.


---

PIC requires no modification to IKE. Instead it uses simplified elements of ISAKMP and IKE to obtain a much less ambitious goal than general IKE, namely the secure provisioning of credentials for successfully authenticated users. The direct use of IKE, e.g. as compared to TLS tunneling [GETCERT], reduces complexity and contributes to the efficiency of the protocol.

The PIC protocol is defined between the Client and the AS. All other exchanges between the entities are implicit in the protocol. This applies in particular to user and machine authentication between the AS and the Back-End Authentication Server, and certification between the AS and the CA.

The four main stages of the proposed PIC protocol are:

1. An optional round of messages provides partial protection of the AS from denial-of-service attacks by verifying that the initiator of the exchange is reachable at the purported source IP address. This is done before any significant CPU or memory resources are consumed by the AS.
2. The protocol establishes a one-way authenticated channel from the client to the AS in which only the server is authenticated.
3. User authentication is performed over this secured channel. User authentication information is transported using EAP tunnelled within ISAKMP.
4. The AS sends the client a (typically short-term) credential, which can be used in subsequent IKE exchanges. This credential can be thought of as a certificate, or a private key generated or stored by the AS and accompanied



by a corresponding certificate. It may also be a symmetric secret key, or other information for deriving such a key.

In stage 4 the created ISAKMP tunnel is used to obtain the secure provisioning of credentials for successfully authenticated users.

Another alternative proposal is HTTP digest authentication within a TLS tunnel, which is not described in detail here.

The above proposals all utilise an inner client (or mutual) authentication protocol within a secure tunnel constructed using a session key resulting from an outer server authentication protocol. In the following description we use AKA as an example of the inner protocol. But the general points of the discussion are applicable to any pair of authentication protocols used in the above manner. This is convenient because the AKA protocol is to be implemented by the USIM of 3G mobile stations, and so this protocol is readily available to a terminal that is to be authenticated. AKA is a challenge-response protocol. In the protocol the "responding entity" which is to be authenticated shares a secret key with the "challenging entity" which is to verify its authenticity. The challenging entity issues to the responding entity a challenge message which includes challenge data, which is typically randomly generated data. The responding entity then generates response data by applying a known function to that challenge data and to the secret key. The response data is returned to the challenging entity in a response message. The challenging entity can also apply the known function to the challenge data and the key, and the result of that can be compared to the response data. Only if the two match is the responding entity is authenticated. In the above proposals AKA is to be used in just the same way as in its pre-existing applications: i.e. the terminal is supplied with challenge data to which it is to apply the function, and then it returns the result to the network for authentication.

In place of the EAP encapsulated AKA protocol any EAP type protocol can be used for authentication of the terminal to network.

Figure 5 illustrates the logical architecture that is common to all of the above proposals. First, a stage-one authentication exchange 40 takes place. In that exchange the identity of a back-end server 41 is authenticated to a terminal entity 42. However, the identity of the terminal entity is not authenticated to the back-end server. Then a stage-two authentication exchange 43 takes place, in which it is intended that the terminal entity is authenticated to the back-end server. Since each of the proposals is intended to be compatible with existing terminal entities, the existing authentication protocol (e.g. EAP/AKA), which has hitherto been usable as a stand-alone protocol, is used for the essential authentication part of stage two. In stage two the back-end server issues a challenge 44 to the terminal entity. The challenge is of such a type that the back-end server can (either by itself or through referral to a further authentication server) authenticate the terminal by means of its response to the challenge. The terminal entity processes the challenge and formulates its response (step 45) and returns that response to the back-end server (step 46). Then the back-end server checks the response (or refers it for checking) (step 47). If the response is correct then the back-end server has authenticated the terminal entity and can issue it with credentials that allow it to carry out subsequent secure transactions with the server (step 48). Otherwise no such credentials are issued and the terminal entity cannot carry out such transactions.

Up to now it has been supposed that the above proposals provide for secure authentication of both the back-end server and the terminal entity to each other. However, the inventors of the present invention have observed that this is not the case.

The vulnerability of the above proposals will be described with reference to figure 6. In figure 6 like items are numbered as for figure 5. In figure 6 a "man-in-the-middle" (MITM) attacker 49, who intends to impersonate a terminal entity 42, takes the place of the terminal entity in figure 5. The MITM performs the stage-one authentication with the back-end server. Then during the stage-two authentication when it receives the challenge message (step 44) from the back-end server it forwards it on to the

terminal entity 42. (Step 50) Using the pre-existing AKA protocol, it returns the appropriate response to the MITM. (Step 51). Then the MITM forwards that response to the back-end server. (step 46). Since the response is the correct response for authentication of terminal entity 42 the back-end server will authenticate the MITM as terminal entity 42 (step 47) and issue it with credentials accordingly (step 48). That may allow the MITM access to supposedly secure services, impersonating the identity of terminal entity 42.

The proposals described above have been developed with the aim of using the existing challenge-response protocol for the essential authentication part of stage two, since that is already a feature of the terminals that are to use the system. Since the proposals fail to provide adequate security,, a new approach is needed.


A method for authenticating a terminal in a communication system, the terminal comprising identification means for applying an authentication function to input data to form response data, and the communication system being arranged to utilise a first authentication protocol for authentication of the terminal, wherein an authentication functionality and the terminal share challenge data, the terminal forms response data by applying the authentication function to the challenge data by means of the identification means, and returns the response data to the authentication functionality, and the authentication functionality authenticates the terminal by means of the response data; the method comprising; executing a second authentication protocol wherein the terminal authenticates the identity of a network entity and the terminal and the network entity share a key for use in securing subsequent communications between the terminal and the network entity; and subsequently executing a third authentication protocol by the steps of: sharing challenge data between the network entity and the terminal; forming at the terminal response data by at least applying the authentication function to the challenge data by means of the identification means; transmitting a response message, comprising the response data, from the terminal to the network entity; and determining based on the response data whether to provide the terminal with access to a service; wherein in the

determining step the terminal is provided with access to the service only if the response data equals a predetermined function of the challenge data and the key.

---

According to a second aspect of the present invention there is provided a communication system comprising a terminal, a network entity and an authentication functionality, the terminal comprising identification means for applying an authentication function to input data to form response data, and the communication system being arranged to utilise a first authentication protocol for authentication of the terminal, wherein an authentication functionality supplies challenge data to the terminal, the terminal forms response data by applying the authentication function to the challenge data by means of the identification means, and returns the response data to the authentication functionality, and the authentication functionality authenticates the terminal by means of the response data; and the communication system being arranged to perform an authentication method comprising the steps of: executing a second authentication protocol wherein the terminal authenticates the identity of a network entity and the terminal and the network entity share a key for use in securing subsequent communications between the terminal and the network entity; and subsequently executing a third authentication protocol by the steps of: transmitting a challenge message of the third protocol, comprising challenge data, from the network entity to the terminal; forming at the terminal response data by at least applying the authentication function to the challenge data by means of the identification means; transmitting a response message, comprising the response data, from the terminal to the network entity; and determining based on the response data whether to provide the terminal with access to a service; wherein in the determining step the terminal is provided with access to the service only if the response data equals a predetermined function of the challenge data and the key.

The predetermined function may be used in a dedicated authentication step. In that case the said message may be a dedicated authentication message. Alternatively, or in addition, the predetermined function may be used for encryption and/or authentication of communications between the terminal and the network entity. In that case, the predetermined function may take an additional data input, such as text



to be encrypted. In that situation access to a service that makes use of that additional data may inherently be denied if the authentication data equals a predetermined function of the first key and the second key since the network entity may be unable to decrypt and/or verify the authenticity of the additional data.

Preferably there are steps of forming test data by applying the authentication function to the challenge data at the authentication functionality; and transmitting the test data from the authentication function to the network entity. Preferably the determining step comprises forming authentication data by applying the predetermined function to the test data and the key at the network entity. Preferably in the determining step the terminal is provided with access to the service only if the response data equals the authentication data.

Preferably there are steps of transmitting the key from the network entity to the authentication functionality; forming test data by applying the authentication function to the challenge data at the authentication functionality; and forming authentication data by applying the predetermined function to the test data and the key at the authentication functionality.

Preferably there are steps of transmitting the result data from the network entity to the authentication functionality; transmitting from the authentication functionality to the network entity an indication of whether the result data equals the authentication data. Preferably in the determining step the terminal is provided with access to the service only if the indication is that the response data equals the authentication data.

Preferably the method comprises transmitting the authentication data from the authentication functionality to the network entity. Preferably in the determining step the terminal is provided with access to the service only if the indication is that the response data equals the authentication data.

Preferably the response data is formed as a cryptographic checksum.

In each case the terminal, and preferably several terminals operating in the system, is/are configured to form the response data by: forming an intermediate value by applying the authentication function to the challenge data by means of the identification means, and forming the response data by applying the predetermined function to the intermediate value and the key as shared with the network entity to which it returns the response message. Whether nested functions are evaluated at the terminal or the authentication functionality, it will be appreciated that the formation of an intermediate or test value may be conceptual, since the nested functions may be evaluated as a single, combined function.

The authentication means may be an identity module of the terminal, or it could be implemented as part of the general processing functionality of the terminal, for example on a general-purpose ASIC of the terminal. The identity module may be user-removable from the terminal. The identity module may be a SIM or a USIM. The first authentication protocol may be the AKA protocol. The first authentication protocol may be used by the system for authenticating a terminal for the purpose of granting or denying it access to the system. In that way, the protocol is likely to be widely or universally supported by terminals operating in the system.

The test data and/or the intermediate value may include one or both of the AKA IK value or the AKA CK value.

Preferably the authentication means stores a code and the authentication function comprises applying a cryptographic transformation to the code and the input data. That code is preferably stored securely by the authentication means and the authentication function.

The network entity may be co-located with the authentication functionality.

The present invention will now be described by way of example with reference to the accompanying drawings.



In the drawings:

figure 1 shows the relationship between EAP client, backend authentication server and NAS in PEAP;

figure 2 shows an architecture overview of EAP SIM GMM authentication;

figure 3 shows the network architectural model for EAP-TTLS;

figure 4 shows relations between PIC entities;

figure 5 shows the logical operation of proposed authentication protocols;

figure 6 illustrates a man-in-the-middle attack;

figure 7 shows an authentication protocol; and

figure 8 illustrates a frustrated man-in-the-middle attack.

In the examples to be described below authentication between two entities is performed in two stages. In stage one a first ("outer") protocol is used to authenticate a back-end server to a terminal entity. In stage two a second ("inner") protocol is used to authenticate the terminal entity to the back-end server. To address the man-in-the-middle attack discussed above, a link is made between the inner and the outer protocols with the aim that only the entity that authenticated the back-end server in stage one can be authenticated by the back-end server in stage two.

The authentication system will be described in more detail with reference to figures 7 and 8. Figure 7 shows the operation of the system for proper authentication of a terminal entity. Figure 8 shows the operation of the system in frustrating a man-in-the-middle attack. It will be appreciated that these are merely examples and that the present invention is applicable to any generic combination of protocols that satisfy the requirements mentioned above: preferably that the inner protocol is used in other contexts too, the outer protocol does only server authentication, and both protocols result in session keys.

The entities shown in figure 7 are a terminal entity 60 (e.g. a 3G UE), a back-end server 61, an AKA authenticator 62 (e.g. a RAS or HSS) and a public key server 63.



The terminal entity is a GSM or 3G terminal incorporating a subscriber identity module (SIM or USIM), which implements the AKA protocol or another like protocol.

---

The terminal entity can also be a data terminal or any other end user device with network connectivity. According to this protocol the SIM or USIM securely stores a secret key  $K$  and when supplied with certain data  $D$  by the terminal can apply a two standard functions  $f$  and  $g$  to that data and the key to return two values, known as  $CK$  and  $IK$ . Thus:

$$CK = f(K,D) \text{ and } IK = g(K,D)$$

To maintain the security of the key the standard function is chosen so that it is very difficult to work out the key from the values that are returned.

The back-end server is a server that is expected to provide the terminal with access to some transaction that calls for authentication. This could, for example, be access to money or confidential data such as e-mails.

The AKA authenticator stores the secret keys of a number of subscribers, each in association with an identification of subscriber information relating to the SIM that stores the respective key. The AKA authenticator also knows functions  $f$  and  $g$ , so it can also calculate  $CK$  and  $IK$  for a given key and a given value of  $D$ .

In figure 7 the following steps take place:

- Step 70 The terminal initiates stage-one authentication by transmitting a stage-one request message (RQ1) to the back-end server.
- Step 71 The back-end server returns a stage-one response message (RS1) which might indicate its public key and one or more certificates. The actual form of this message depends on the protocol used.
- Step 72 The terminal transmits data to the public key server 63 to verify the identity of the back-end server. This may involve requesting a revocation list from

the public key server 63, or requesting an on-line status check by supplying the serial number of a certificate received in step 71.

Step 73 The public key server performs the appropriate check based on the data supplied in step 72, for instance by looking up the serial number received from the terminal in its database to determine the entity with which it is associated.

Step 74 The public key server returns the appropriate requested information to the terminal.

Step 75 The terminal verifies that the entity indicated in the message at step 74 is the one with which it intends to communicate, using the information received from the public key server. If so, it begins a secure communication session with the back-end server using the public key. Otherwise, it terminates the process.

Step 76 If the back-end server has been authenticated the back-end server and the terminal share a session key T. This is typically generated by both parties independently, from the authentication keys of the network entity and from the data exchanged in the protocol

Step 77 The back-end server initiates stage-two authentication by transmitting an EAP – Request/Identity message to the terminal.

At this stage, the back-end server is authenticated to the terminal, and the back-end server and the terminal share session key T. In this example the session key is a PIC SA (server authentication) key, and subsequent messaging is done using the PIC protocol, secured using that key. However, other protocols could be used.

Step 78 The terminal transmits to the back-end server its identity (IMSI) by means of an EAP – Response / Identity message encapsulated by PIC.

Step 79 The back-end server sends the terminal's identity to the AKA authenticator 62. For this step a MAP (Send\_Auth\_Params: IMSI) message or a DIAMETER message could be used.

---

Step 80 The AKA authenticator looks up the secret key  $K$  it stores for the terminal, generates a random number  $RAND$  and applies the standard functions  $f$  and  $g$  to determine  $CK = f(RAND, K)$  and  $IK = g(RAND, K)$ .

Step 81 The AKA authenticator transmits to the back-end server the usual AKA quintuplet comprising  $RAND$ ,  $AUTN$ ,  $RES$ ,  $CK$  and  $IK$ . This may be sent as a MAP (AKA authentication quintuplet) message.

Step 82 The back-end server transmits a challenge message including  $RAND$  and  $AUTN$  to the terminal.  $AUTN$  is supplied so as to authorise the terminal to carry out the operation at step 83. This message may be transmitted as an EAP – Request / AKA-challenge ( $RAND$ ,  $AUTN$ ) message encapsulated by PIC.

Step 83 The terminal applies the received  $RAND$  to its SIM or USIM 65, which returns the corresponding  $SRES$ ,  $CK$  and  $IK$ .

Subsequently the terminal returns the value  $SRES$  to the back-end server. This message may be transmitted as an EAP – Response / AKA-challenge ( $RES$ ) message encapsulated by PIC.

Step 84 The terminal applies a further standard function  $h$ , which is known to the back-end server to the  $IK$  returned by the SIM and the session key  $T$  obtained during the stage-one authentication, to generate a checksum (or result) value  $C$ :

$$K = h(IK, T, \text{text}), C = \text{MAC}(K, \text{CREDENTIAL-REQUEST})$$

Thus a new key  $K$  (common to all the present methods) and then a checksum  $C$  are calculated. MAC is a message authentication code: suitable algorithm is e.g., HMAC\_SHA1.

The function  $h$  is chosen so that it is infeasible to compute  $K$  without the knowledge of both  $IK$  and  $T$ . An example of function  $h$  is: SHA-1, or HMAC-SHA-1 with key  $IK$  and a data field comprising of  $T$  and text. The data text may be any session related data known to the terminal and to the back-end server. Steps 84 to 87 are mandated as an integral part of the protocol, with the result that the terminal has to perform them as laid down in the protocol. For example, suitable steps can be added as mandatory steps to PIC, PEAP and other similar protocols to make them secure.

Step 85 The terminal returns the value  $K$  to the back-end server. This message may be transmitted encapsulated by PIC. In addition, the terminal also transmits a CREDENTIAL-REQUEST which is a field in the PIC message and a checksum  $C$ , which may also be transmitted as a field in the PIC message

Step 86 The back-end server, which also has knowledge of function  $h$ , as well as the keys  $IK$ , received from the AKA authenticator functionality (AA authenticating server), and the terminal ( $T$ ), computes:

$K' = h(IK, T)$ ,  $C' = \text{MAC}(K, \text{CREDENTIAL-REQUEST})$   
using the value of  $IK$  supplied to it by the AKA authenticator.

Step 87 The back-end server compares the value of  $K$  returned by the terminal with the value of  $K'$  it has received from AA (this comparison of RES is standard part of the AKA protocol). The back-end server also compares  $C$  with the value  $C'$  it has calculated itself. If and only if the two match, then the CREDENTIAL-REQUEST allegedly sent by the terminal is deemed to be authenticated to the back-end server.

Step 88 If the terminal is authenticated to the back-end server then the back-end server transmits a security credential CERT to the terminal. Using that credential the terminal may then perform a secure transaction with the back-end server. This message may be transmitted as a field in a PIC message .

---

It is to be noted that in this system the value that is returned by the terminal to the back-end server as the response in the stage-two challenge-response protocol is dependent on the session key shared between the terminal and the back-end server in stage one.

The procedure described above is an example of Method 2 to be described later.

Figure 8 shows what happens in this system if a man-in-the-middle attack is attempted. In figure 8 the terminal entity 60, back-end server 61, AKA authenticator 62 and public key authenticator 63 are as for figure 7. In addition, figure 8 shows an MITM 64 who is attempting to assume the identity of terminal 60.

In figure 8 steps 90 to 97 are the same as steps 70 to 77 respectively in figure 7, but with the MITM 64 taking the place of terminal 60.

Then the following steps take place as the attack proceeds:

Step 98 The MITM passes the request for the IMSI on to the terminal 60.

Step 99 The terminal returns its IMSI to the MITM.

Step 100 The MITM passes the IMSI on to the back-end server.

Step 101 The back-end server passes the IMSI on to the AKA authenticator.

Step 102 The AKA authenticator determines an AKA quintuplet for the supplied IMSI.

Step 103 The AKA authenticator passes the AKA quintuplet to the back-end server.

Step 104 The back-end server passes RAND and AUTN to the MITM.

Step 105 The MITM passes RAND and AUTN to the terminal 60.

Step 106 The terminal determines IK for the supplied RAND.

Step 107 If it the MITM has not established a shared key with the terminal 60 then the process will fail at this point because the terminal has no value of T to supply to function h. Alternatively, if the MITM has established a shared key (T') then the terminal 60 can calculate:

$$K^* = h(IK, T'), C^* = \text{MAC}(K^*, \text{CREDENTIAL-REQUEST})$$

However, this key T' will with high probability be different from the key T that is shared by the MITM and the back-end server. This is because it is necessarily impossible for a peer entity in the protocol to run two instances of the protocol with two different peer entities in such a manner that the values for parameter T / T' in both protocols are equal with non-negligible probability. All commonly used protocols for network authentication (TLS and the protocol designed for PIC) are such that neither party can force the values of the session keys. This is often stated explicitly as a security property of the protocol. However, it should be noted this requirement not essential because even if the MITM succeeds in convincing the victim to use the same key T, the victim will compute C using its *own* CREDENTIAL-REQUEST. So the MITM will still not be able to gain access (see below) because the goal of MITM is to send his own CREDENTIAL-REQUEST to the back-end server and pretend that it came from the client. Since the MITM does not know IK, he cannot compute K and hence he cannot compute a valid checksum for a different CREDENTIAL-REQUEST. The exact form of T and its content should suitably be specified for each stage 1 protocol separately bearing in mind that it should be impossible to force the terminal to a predetermined T. This is typically easy to achieve. Moreover, the text field can be defined in such a way that K and K\* are necessarily different.

Step 108 If the process has not already failed, the terminal 60 passes  $K^*$  to the MITM. In addition, the terminal also transmits a CREDENTIAL-REQUEST which is a field in the PIC message and a checksum C, which may also be transmitted as a field in the PIC message.

---

Step 109 The MITM passes  $K^*$  to the back-end server. In addition, the MITM also forwards the CREDENTIAL-REQUEST which is a field in the PIC message and the checksum C, which may also be transmitted as a field in the PIC message

Step 110 The back-end server computes  $K'=h(IK,T,\text{text})$  using the value of IK supplied to it by the AKA authenticator.

Step 111 Because of the definition of T and the text field, and since h is a cryptographically secure function it is very unlikely that  $K^*$  will equal  $K'$ . Therefore, the MITM is not authenticated to the back-end server.

Thus the previously-discussed man-in-the-middle attack is prevented.

The location at which some of the processes are performed could be different from that described above. Some examples are given in methods 1 to 3 below.

The back-end server could securely supply the credential to another unit with which the terminal could then conduct a secured transaction. In that case the back-end server could simply act as a terminal authentication server. The back-end server and the AKA authenticator or other authentication functionality could be integrated into one unit.

The present invention is not limited to operation with any specific type of communication system. However, the system is preferably a mobile communication system such as GSM or the third-generation (3G) system. Instead of the AKA

protocol another authentication protocol supported by the terminal could be used. This is preferably the protocol used for granting or denying a terminal access to the network. The terminal is preferably capable of wireless (e.g. radio) communication with the back-end server (network entity) and the authentication server. The terminal may be a mobile station of the communication system.

The invention provides several different methods to provide authentication of the entity that holds the client's copy of the session keys and credentials. The presented solution is applicable to mutual authentication protocols that are constructed as a combination of two unilateral authentications, including PEAP, EAP-TTLS and PIC.

In the following further examples we denote by  $K$  the session master key. In PEAP and EAP-TTLS the master key  $K$  is the TLS master key that is sent to the network access point (or access server), which uses  $K$  to derive the session keys for the wireless link. In PIC the key  $K$  is the key that is used to transport (e.g.) IKE credentials to the client terminal (C). The problem discovered above is due to the fact that in the prior proposed protocols  $K$  depends solely on the same secrets as the protection tunnel, and is therefore susceptible of being known to any peer at the end-point of the tunnel.

Further, let us denote by  $S$  some secrets derived by the C and the RAS (in the same role as AA above) in the second authentication protocol. The exact requirements for  $S$  are given later. Let  $T$  denote the master key that is used to derive the secret keys for the protection tunnel. Hence  $S$  is a secret shared by C and the RAS and  $T$  is a secret shared by C and AS (which is in the same role as BE above). For example, the TLS master key derived in the TLS handshake of PEAP is a typical example of  $T$ . In the previous solutions the secret  $S$  is not used (thereby allowing the man-in-the-middle attack). The master link key derived in EAP/SIM or EAP/AKA satisfies the requirements for  $S$  and can be used for this purpose. There are two ways of using  $S$  to achieve the necessary binding (or linking) between  $S$  and  $K$ . In the first method the binding is established directly by taking  $S$  as the input to the computation of



session key  $K$ . The other two methods make use of a cryptographic check value to verify that the client  $C$  who is in possession of  $T$  is also in possession of  $S$ .

---

Method 1: After the remote authentication protocol between the  $C$  and the RAS is successfully completed the RAS sends the derived secret  $S$  to the AS (AKA authentication functionality) through a protected (Diameter) connection provided by the network infrastructure. The AS uses it, in addition to  $T$ , as input to the derivation of  $K$ . The client  $C$  knows  $S$  and  $T$  and performs identical computations to derive its copy of  $K$ .

Method 2: After the  $C$  and the AS have established the protection tunnel and the remote authentication protocol between the  $C$  and the RAS is successfully completed, the AS sends  $T$  to the RAS over a protected (e.g. Diameter) connection. The RAS computes a verification value  $V$  (a cryptographic check-sum) from  $T$  and  $S$ . The client  $C$  uses its values of  $T$  and  $S$  and the same check-sum computation method to compute its value of  $V$ . Then the  $C$  sends its value of  $V$  to the AS. The comparison of the two values of  $V$  is now performed by the AS or the RAS. If the values are compared by the AS, then the RAS sends its value of  $V$  to the AS. Another possibility is that the AS sends the value of  $V$  received from the  $C$  forward to the RAS, which performs the comparison. In this case the RAS needs to communicate the result of the comparison to the AS. If the values are equal, the AS derives  $K$  based on  $T$ . The client  $C$  uses its own copy of  $T$  to derive its session keys  $K$ .

Method 3: After the  $C$  and the AS have established the protection tunnel and the remote authentication protocol between the  $C$  and the RAS is successfully completed, the RAS sends  $S$  to the AS over a protected (e.g. Diameter) connection. The AS computes a verification value of  $V$  (cryptographic check-sum, e.g. TLS MAC) from  $T$  and  $S$ . The client  $C$  uses its values of  $T$  and  $S$  and the same check-sum computation method to compute its value of  $V$ . Then the  $C$  sends its value of  $V$  to the AS. The comparison of the two values of  $V$  is now performed by the AS. If the

values are equal, the AS derives K based on T. The client C uses its own copy of T to derive its session keys K.

The three methods described above do not require any changes to the remote client authentication protocol. The support for the derivation of S can most naturally be added in the EAP encapsulation of the protocol in a similar way that current EAP encapsulations of existing authentication protocols provide methods for session key derivation. In Methods 1 and 3 the secret S is sent from the RAS to the AS using a protected channel, which depends on the trust relationships of the network infrastructure. Therefore the usage of S for remote authentication in protected mode must not endanger the original use of the remote authentication protocol. This is achieved if a one-way function is used to compute S from the original session keys.

Methods 2 and 3 do not require any changes to the methods by which the session key K is derived from T. Of these two methods, Method 3 may have some advantage over Method 2 due to the fact that the protection tunnel may provide check-value (TLS MAC or IPsec MAC) computation algorithms in the C and the AS.

Method 1 requires support for the usage of S as additional secret parameter in the derivation of the session key K. However, this is easy to achieve.

Further details of the protocols mentioned above are available as follows:

PIC: Y. Sheffer, H. Krawczyk, Bernard Aboba, PIC, A Pre-IKE Credential Provisioning Protocol; <draft-ietf-ipsra-pic-05.txt> 11 February 2002

PEAP: H. Andersson, S. Josefsson, Glen Zorn, Dan Simon, Ashwin Palekar, Protected EAP Protocol (PEAP); <draft-josefsson-pppext-eap-tls-eap-02.txt> 23 February 2002

TTLS: Paul Funk Simon Blake-Wilson, EAP Tunneled TLS Authentication Protocol (EAP-TTLS); <draft-ietf-pppext-eap-ttls-01.txt> February 2002

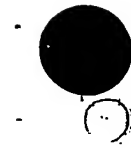
GETCERT: Bellovin, S., and Moskowitz, B., "Client Certificate and Key Retrieval for IKE", Internet draft (work in progress), draft-bellovin-ipsra-getcert-00.txt, June 2000.

The following abbreviations are used herein:

---

SIM= Subscriber Identity Module  
AKA= Authentication and Key Agreement  
OWLAN= Operator Wireless Local Area Network  
PPP= Point-to-Point Protocol  
IEEE= The Institute of Electrical and Electronics Engineers  
USIM= Universal Subscriber Identity Module  
IETF=Internet Engineering Task Force  
PEAP= Protected EAP Protocol  
TLS= Transport Layer Security  
TTLS=Tunneled TTLS  
pppext="Point-to-Point protocol extensions" working group  
PIC= A Pre-IKE Credential Provisioning Protocol  
ISAKMP=Internet security association and key management protocol  
AAA= authentication, authorization, and accounting  
IPsec=Internet protocol security  
RADIUS=Remote Authentication Dial In User Service  
IKE=Internet key exchange  
GETCERT=Internet draft "Client Certificate and Key Retrieval for IKE"  
CPU= Central Processing Unit  
HTTP= Hyper Text Transfer Protocol  
ASIC= application-specific integrated circuit  
GMM=GPRS Mobility Management  
GPRS=General packet radio service  
RAS= registration, admission and status  
HSS= Home Subscriber Server  
IMSI=International Mobile Subscriber Identity  
AUTN=Authentication Token

The applicant hereby discloses in isolation each individual feature described herein and any combination of two or more such features, to the extent that such features or



combinations are capable of being carried out based on the present specification as a whole in the light of the common general knowledge of a person skilled in the art, irrespective of whether such features or combinations of features solve any problems disclosed herein, and without limitation to the scope of the claims. The applicant indicates that aspects of the present invention may consist of any such individual feature or combination of features. In view of the foregoing description it will be evident to a person skilled in the art that various modifications may be made within the scope of the invention.

## CLAIMS

---

1. A method for authenticating a terminal in a communication system, the terminal comprising identification means for applying authentication functions to input data to form response data, and the communication system being arranged to utilise a first authentication protocol for authentication of the terminal, wherein an authentication functionality and the terminal share challenge data, the terminal forms response data and a first key by applying the authentication functions to the challenge data by means of the identification means, and returns the response data to the authentication functionality, and the authentication functionality authenticates the terminal by means of the response data and can apply an authentication function to the challenge data to duplicate the first key; the method comprising;

executing a second authentication protocol wherein the terminal authenticates the identity of a network entity and the terminal and the network entity share a second key for use in securing subsequent communications between the terminal and the network entity;

and subsequently executing a third authentication protocol by the steps of:

sharing challenge data between the network entity and the terminal;

forming at the terminal a first key by at least applying one of the authentication functions to the challenge data by means of the identification means;

transmitting a message comprising authentication data, from the terminal to the network entity;


and determining based on the authentication data whether to provide the terminal with access to a service;

wherein in the determining step the terminal is provided with access to the service only if the authentication data equals a predetermined function of at least the first key and the second key.

2. A method as claimed in claim 1, wherein the method comprises:

forming test data by applying the authentication function to the challenge data at the authentication functionality; and

transmitting the test data from the authentication function to the network entity;



and wherein the determining step comprises forming authentication data by applying the predetermined function to the test data and the key at the network entity; and in the determining step the terminal is provided with access to the service only if the response data equals the authentication data.

3. A method as claimed in claim 1, wherein the method comprises:

- transmitting the key from the network entity to the authentication functionality;
- forming test data by applying the authentication function to the challenge data at the authentication functionality; and
- forming authentication data by applying the predetermined function to the test data and the key at the authentication functionality.

4. A method as claimed in claim 3, comprising:

- transmitting the result data from the network entity to the authentication functionality;
- transmitting from the authentication functionality to the network entity an indication of whether the result data equals the authentication data;

and wherein in the determining step the terminal is provided with access to the service only if the indication is that the response data equals the authentication data.

5. A method as claimed in claim 3, comprising:

- transmitting the authentication data from the authentication functionality to the network entity;

and wherein in the determining step the terminal is provided with access to the service only if the indication is that the response data equals the authentication data.

6. A method as claimed in any preceding claim, wherein the response data is formed as a cryptographic checksum

7. A method as claimed in any preceding claim, wherein the network entity is co-located with the authentication functionality.

8. A method as claimed in any preceding claim, wherein authentication means is an identity module of the terminal.

---

9. A method as claimed in claim 8, wherein the identity module is user-removable from the terminal.

10. A method as claimed in claim 8 or 9, wherein the identity module is a SIM or a USIM.

11. A method as claimed in any preceding claim, wherein the first authentication protocol is the AKA protocol or any protocol of the EAP family.

13. A method as claimed in claim 11 as dependent on any of claims 2 to 6, wherein the test data includes one or both of the AKA IK value or the AKA CK value.

14. A method as claimed in any preceding claim, wherein the authentication means stores a code and the authentication function comprises applying a cryptographic transformation to the code and the input data.

15. A method as claimed in any preceding claim, wherein the second authentication protocol is the PIC, the PEAP protocol or the EAP-TTLS protocol.

16. A method as claimed in any preceding claim, wherein the challenge message and the response message are formed according to the EAP protocol.

17. A communication system comprising identification means for applying authentication functions to input data to form response data, and the communication system being arranged to utilise a first authentication protocol for authentication of the terminal, wherein an authentication functionality and the terminal share challenge data, the terminal forms response data and a first key by applying the authentication functions to the challenge data by means of the identification means, and returns the response data to the authentication functionality, and the authentication functionality

authenticates the terminal by means of the response data and can apply an authentication function to the challenge data to duplicate the first key; the system being arranged to perform an authentication method comprising the steps of:

executing a second authentication protocol wherein the terminal authenticates the identity of a network entity and the terminal and the network entity share a second key for use in securing subsequent communications between the terminal and the network entity;

and subsequently executing a third authentication protocol by the steps of:

sharing challenge data between the network entity and the terminal;

forming at the terminal a first key by at least applying one of the authentication functions to the challenge data by means of the identification means;

transmitting a message comprising authentication data, from the terminal to the network entity;

and determining based on the authentication data whether to provide the terminal with access to a service;


wherein in the determining step the terminal is provided with access to the service only if the authentication data equals a predetermined function of at least the first key and the second key.

18. A communication system comprising a terminal, a network entity and an authentication functionality, the terminal comprising identification means for applying an authentication function to input data to form response data, and the communication system being arranged to utilise a first authentication protocol wherein the terminal authenticates the identity of a network entity and the terminal and the network entity share a key for use in securing subsequent communications between the terminal and the network entity; and the communication system being arranged to perform an authentication method comprising the steps of: executing a second authentication protocol for authentication of the terminal, wherein an authentication functionality supplies challenge data to the terminal, the terminal forms response data and test data by applying the authentication function to the challenge data by means of the identification means, and returns the response data to the authentication functionality, and the authentication functionality authenticates the



terminal by means of the response data; and subsequently executing a third linking protocol by the steps of forming at the terminal secret session keys by at least applying a predetermined function to the secret test data by means of the shared key established in the first protocol; forming at the network entity secret session keys by at least applying a predetermined function to the secret test data by means of the shared key established in the first protocol; wherein in the secret session keys are used to secure the subsequent communication between the terminal and some network element.

19. A an authentication method for use in a communication system comprising a terminal, a network entity and an authentication functionality, the terminal comprising identification means for applying an authentication function to input data to form response data, and the communication system being arranged to utilise a first authentication protocol wherein the terminal authenticates the identity of a network entity and the terminal and the network entity share a key for use in securing subsequent communications between the terminal and the network entity; and the authentication method comprising the steps of: executing a second authentication protocol for authentication of the terminal, wherein an authentication functionality supplies challenge data to the terminal, the terminal forms response data and test data by applying the authentication function to the challenge data by means of the identification means, and returns the response data to the authentication functionality, and the authentication functionality authenticates the terminal by means of the response data; and subsequently executing a third linking protocol by the steps of forming at the terminal secret session keys by at least applying a predetermined function to the secret test data by means of the shared key established in the first protocol; forming at the network entity secret session keys by at least applying a predetermined function to the secret test data by means of the shared key established in the first protocol; wherein in the secret session keys are used to secure the subsequent communication between the terminal and some network element.



20. A method for authenticating a terminal in a communication system, substantially as herein described with reference to the figures 7 and 8 of the accompanying drawings.

21. A communication system substantially as herein described with reference to figures 7 and 8 of the accompanying drawings.

## ABSTRACT

---

### LINKED AUTHENTICATION PROTOCOLS

A method for authenticating a terminal in a communication system, the terminal comprising identification means for applying authentication functions to input data to form response data, and the communication system being arranged to utilise a first authentication protocol for authentication of the terminal, wherein an authentication functionality and the terminal share challenge data, the terminal forms response data and a first key by applying the authentication functions to the challenge data by means of the identification means, and returns the response data to the authentication functionality, and the authentication functionality authenticates the terminal by means of the response data and can apply an authentication function to the challenge data to duplicate the first key; the method comprising; executing a second authentication protocol wherein the terminal authenticates the identity of a network entity and the terminal and the network entity share a second key for use in securing subsequent communications between the terminal and the network entity; and subsequently executing a third authentication protocol by the steps of: sharing challenge data between the network entity and the terminal; forming at the terminal a first key by at least applying one of the authentication functions to the challenge data by means of the identification means; transmitting a message comprising authentication data, from the terminal to the network entity; and determining based on the authentication data whether to provide the terminal with access to a service; wherein in the determining step the terminal is provided with access to the service only if the authentication data equals a predetermined function of at least the first key and the second key.

Figure 7



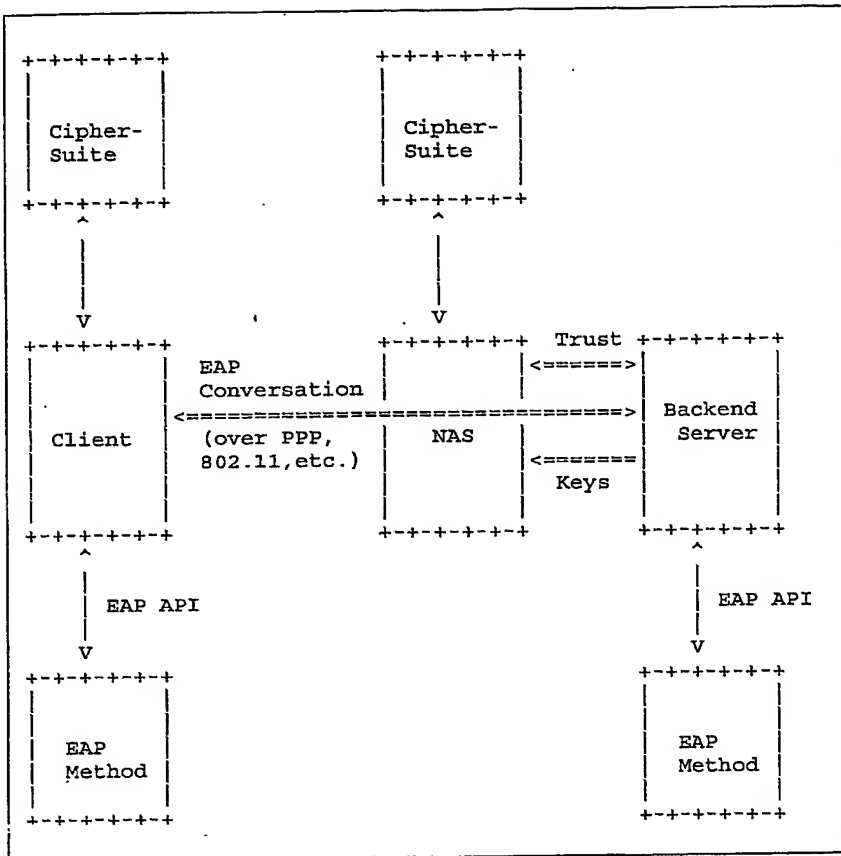


Figure 1 - Relationship between EAP client, backend authentication server and NAS in PEAP



---

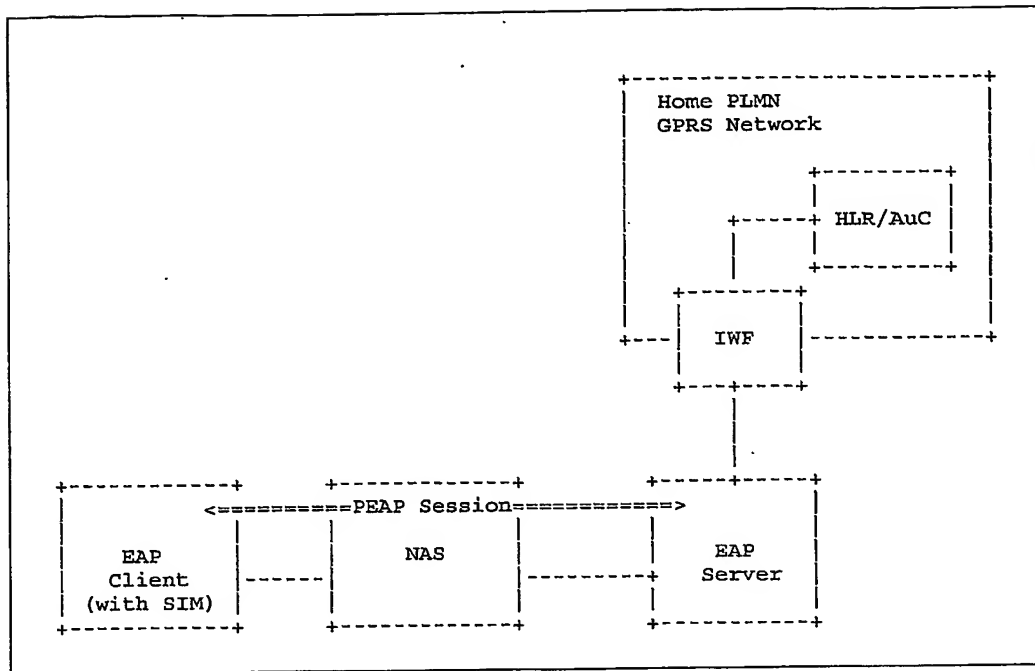
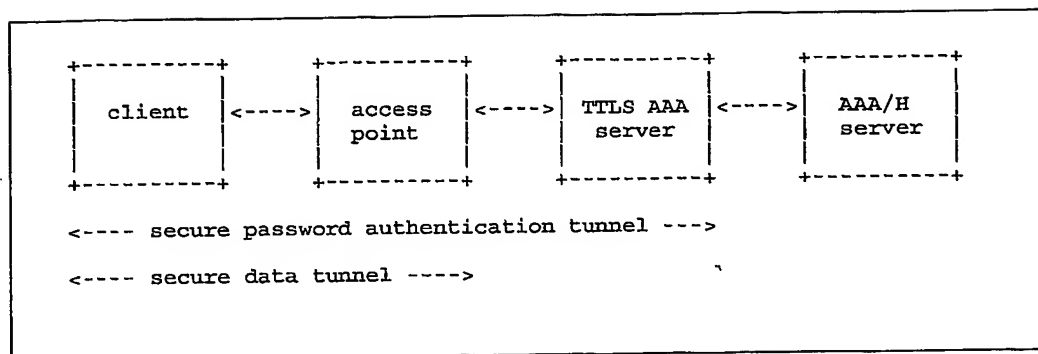


Figure 2 - EAP SIM GMM Authentication: Architecture Overview



**Figure 3 The network architectural model for EAP-TTLS**

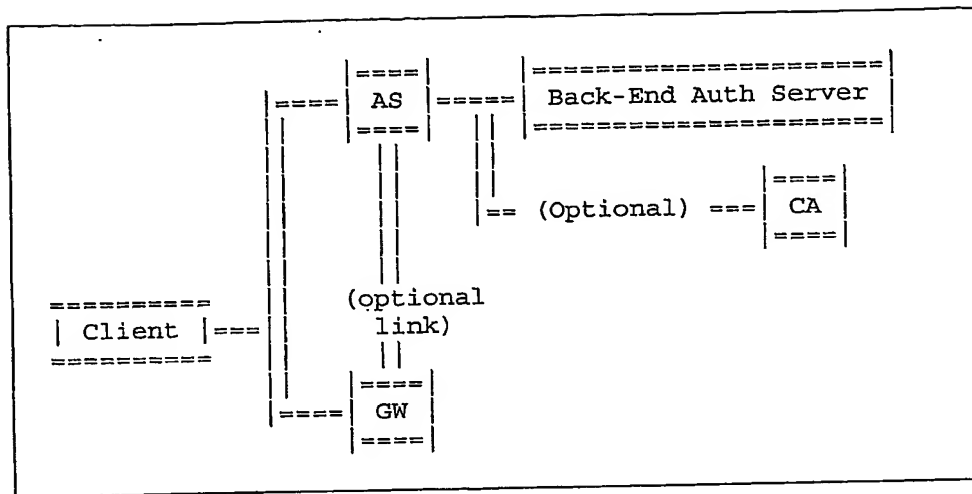


Figure 4 - Relations between the PIC entities



Fig. 5

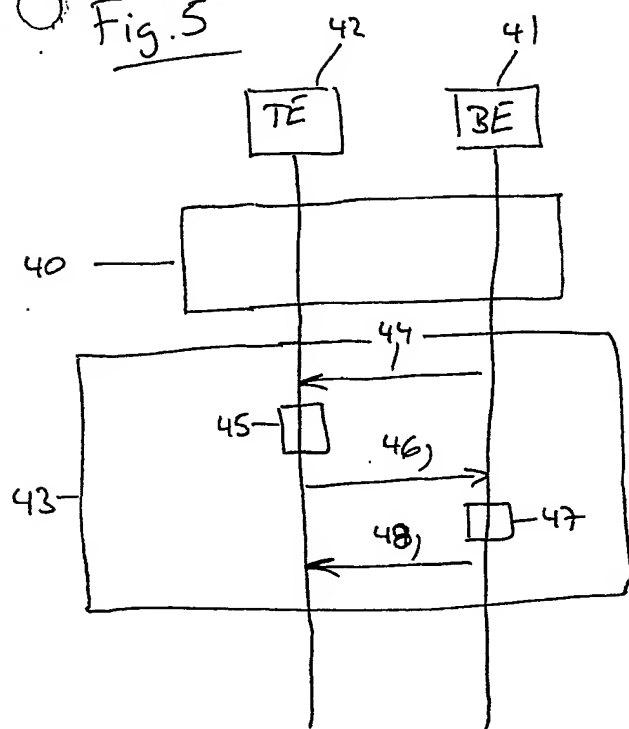


Fig. 6

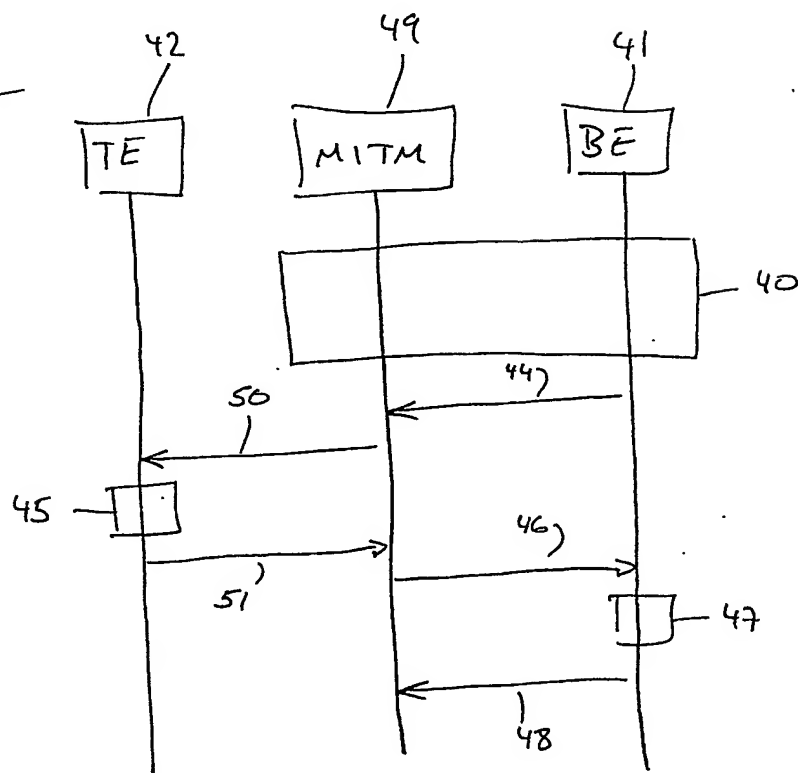


Fig. 7

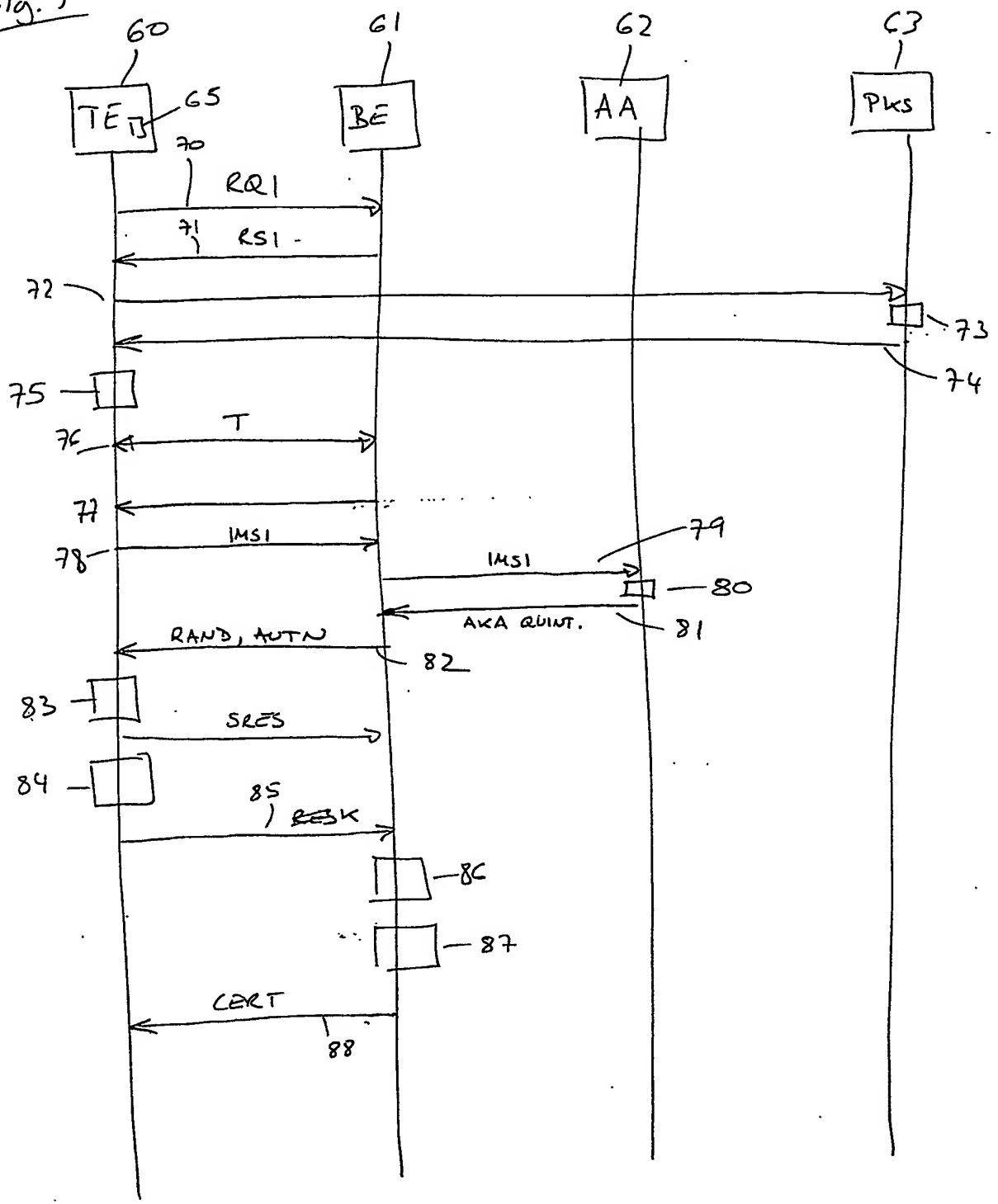


Fig. 8

